REMARKS

Applicants thank the Examiner for the very thorough consideration given the present application. Claims 1-31 remain in the application and claims 7, 19, 25, 27, 29 and 31 are independent.

The Office Action dated June 23, 2009 has been received and carefully reviewed. Each issue raised in the Office Action is addressed below. Reconsideration and Allowance of the present application are respectfully requested in view of the following remarks.

Examiner Interview

Responsive to receipt of the outstanding Office Action, Applicants requested an interview with Examiner Robert McCarry, Jr. to discuss the rejections of record. Applicants and Applicants' representative Paul T. Sewell wish to thank Examiner McCarry, Jr. for the courtesies extended to Applicants' representative during the telephone interview, which was conducted on September 9, 2009. The Interview covered a discussion of the feed forward control of the invention, a discussion of the term "genetic algorithm", a discussion of the applied references and the manner in which they were applied to the claims of record. We also thanked Examiner McCarry for the prior indications of allowability.

The features of the invention were explained, especially with respect to a 1-dimensional running route setting Xj values, each of which have a current target deviation $\Delta R^*(Xj)$ received from data transmission request signal 27-1. The features of the applied prior art were discussed in detail, including Sadano et al, which teaches a vehicle steering control that controls power steering torque as a function of sensed steering angle, yaw angle, vehicle speed, relative side deviation y (at a forward point) and road curvature. Sadano does not have route setting values Xj and does not have a current target deviation. More specifically, Sadano does not have a target deviation from a running route at every current position of the vehicle. Tashiro et al teaches power control of engine output power or torque or a similar transmission control. The Examiner refers to column 2, lines 9-25, and says that this portion is for controlling steering control and "one would reasonably interpret Tashiro to include target steering angle and actual steering angle." Tashiro provides no disclosure of feedback control for a steering system. Tashiro is

CG/PTS/py

limited to power and transmission controls because these systems don't "require all the controlled elements to be redesigned", column 1, lines 33-34, as would occur if one were to attempt to incorporate such a system into a steering control. Even if it taught steering control, which it does not, it does not teach current target deviation, nor optimization to control vibration. Murata et al teaches a variable gear ratio power steering device with what appears to be conventional front end steering structure.

At the conclusion of the Interview the Examiner indicated that he understood the invention disclosed, and he tentatively indicated that based upon our discussion and explanation of the invention, that the applied prior art fails to show or suggest the features of claims 7, 19 and 29, subject to his review of the references. The Examiner indicated that he had also discussed the prior art with several other Examiners and his Supervisor, and already is of the opinion that the best art is already in the record, subject to merely updating the search. He was concerned about the breadth of claim 31 and requested consideration of amending to indicate the contact detection was detected by a contact detection sensor. Applicants and Applicants' representative greatly appreciate Examiner McCarry's willingness to discuss the disclosed invention and the merits of the rejection at length, as well as his preliminary indication that most of the independent claims of record appear to avoid the applied prior subject to a final review. The above constitutes Applicants' statement of the substance of the Interview.

Allowable Subject Matter

Applicants appreciate the Examiner's indication that claims 9, 10, 14-18 and 20-24 are objected to as containing allowable subject matter subject to rewriting in independent form, and his indication that claims 25-28 are allowed. Applicants respectfully request reconsideration of the remaining claims.

Information Disclosure Statement

Applicants again note it appears that some of the references supplied with the Information Disclosure Statement filed along with the application and the second IDS filed December 26, 2006, filed responsive to the European Search Report dated October 27, 2006,

appear to have been lined out as if they were not considered. Applicants believe that the references cited therein were in compliance with all of the requirements of 37 C.F.R. § 1.97 and § 1.98 and therefore again request that the Examiner either consider the remaining references and return the appropriately marked SB 08 forms with the next Office Action, or indicate the basis for any refusal to consider the references so that Applicants may respond appropriately.

Claim Rejections – 35 U.S.C. § 112, Second Paragraph

Claims 11 and 12 stand rejected under 35 U.S.C. § 112, second paragraph, for being indefinite because of the term "genetic algorithm." This rejection is respectfully traversed. As was discussed in the Interview, the term "genetic algorithm" is disclosed in the specification as filed, for example on pages 26-29, and merely refers to a computing technique for optimizing solutions. Such computer simulations were developed in the 1950s and became widely known in the 1960s and early 1970s, as can be confirmed by reference to the 14 page article on the "genetic algorithm" available at Wikipedia, a copy of which is attached hereto. The article provides a further explanation of the manner in which such optimizations, such as to avoid vibrations in this case, may be carried out. It is submitted therefore, as was agreed in the interview, that this is a term known to those skilled in the art of optimization calculations. Therefore, it is respectfully submitted that claims 11 and 12 particularly point out and distinctly claim the subject matter which the Applicants regard as their invention. Reconsideration and withdrawal of the rejection are respectfully requested.

Claim Rejections – 35 U.S.C. § 103

Claims 1-8, 11-13 and 29-31 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Sadano in view of Tashiro. Applicant submits the Examiner has failed to establish a *prima facie* case of obviousness and respectfully traverses the rejection. A complete discussion of the Examiner's rejection is set forth in the Office Action, and is not being repeated here.

In order to establish a *prima facie* case of obviousness under 35 U.S.C. § 103(a), the cited references must teach or suggest each and every element in the claims. See MPEP § 706.02(j); MPEP §§ 2141-2144.

As was discussed in the Interview, the Office Action describes some of the features of Sadano, but fails to show or suggest the features, at least including those recited in lines 8-11 and 13-16 of claim 7. More specifically, lines 8-11 require that the steering control section generates a provisional steering angle based on a current target deviation from a running route at a current position of said vehicle, a current actual deviation from said running route at said current position of said vehicle, and a current target steering angle at said current position of said vehicle, and lines 13-16 requires an optimization calculating section configured to convert said provisional target steering angle to a target control steering angle by adding a correction steering angle based on a steering angle prediction correction to minimize vibration of said vehicle resulting from a steering of said vehicle. Finally, claim 7 requires that the drive section mechanically steers said cart based on said target control steering angle. To the contrary, the system of Sadano does not mechanically steer the car at all, as the driver steers the car, and the system merely controls power to the steering motor to control steering torque. See column 2, lines 65-67, column 6, lines 45-47, and column 7, lines 31-34. Sadano merely senses the steering angle, yaw angle, vehicle speed, side deviation and road curvature, and from that information calculates a steering angle dependent upon steering speed. See column 3, lines 27-45. But Sadano does not show or suggest the use of a "current target deviation from a running route at a current position" of the vehicle, nor does Sadano generate a provisional steering angle based on a current target deviation from a running route at a current position of said vehicle, a current actual deviation from said running route at said current position of said vehicle, and a current target steering angle at said current position of said vehicle as set forth in claim 7. In addition Sadano also does not have an optimization calculating section configured to convert said provisional target steering angle to a target control steering angle by adding a correction steering angle based on a steering angle prediction correction to minimize vibration of said vehicle resulting from a steering of said vehicle as also set forth in claim 7. The result of these features in claim 7 is that the drive section mechanically steers the cart based on the target control steering angle. The result in Sadano is that the driver steers the car with an effort more appropriate to the circumstances.

Similarly, as discussed in the Interview, Tashiro does not show or suggest a steering

CG/PTS/py

feedback system, as the disclosure of Tashiro is limited to feedback systems for engine power or automatic transmission control. See column 1, lines 31-34, as compared to column 2, lines 45-52. Tashiro fails to show or suggest the use of a "current target deviation from a running route at a current position" of the vehicle, nor does Tashiro generate a provisional steering angle based on a current target deviation from a running route at a current position of said vehicle, a current actual deviation from said running route at said current position of said vehicle, and a current target steering angle at said current position of said vehicle as set forth in claim 7. In addition Tashiro also does not have an optimization calculating section configured to convert said provisional target steering angle to a target control steering angle by adding a correction steering angle based on a steering angle prediction correction to minimize vibration of said vehicle resulting from a steering of said vehicle as also set forth in claim 7. Therefore, Tashiro cannot remedy the defects of Sadano as discussed above. With regard to dependent claims 1-6, 8 and 11-13, Applicants submit that claims 1-6, 8 and 11-13 depend, either directly or indirectly, from independent claim 7 which is allowable for the reasons set forth above, and therefore claims 1-6, 8 and 11-13 are allowable based on their dependence from claim 7. Reconsideration and allowance thereof are respectfully requested.

Claim 29 is directed to a method of steering a vehicle guided along a running route without contact with a guide rail, including the steps of setting a 1-dimensional coordinate data of a target route comprising a sequence of position data, setting a target steering angle corresponding only to the 1-dimensional coordinate data, detecting a current deviation between the target routes and the current position of a vehicle main body, and generating a control steering angle corresponding to said current deviation said target steering angle, and turning the wheels to control the steering angle. As was discussed above, and in the Interview, neither Sadano nor Tashiro control the steering wheels, and neither reference shows or suggests setting a target steering angle corresponding only to the 1-dimensional coordinate data. Sadano calculates a target steering angle to control the steering motor current based upon sensing the steering angle, yaw angle, vehicle speed, side deviation and road curvature, all of which are based upon 3-dimensional coordinate data that results from video analysis. See column 3, lines 27-39. Tashiro performs neither of these functions and therefore cannot remedy the defects of Sadano.

With respect to dependent claim 30, Applicants submit that claim 30 depends from independent claim 29 which is allowable for the reasons set forth above, and therefore claim 30 is allowable based on its dependence from claim 29. Reconsideration and allowance are respectfully requested.

While not conceding the appropriateness of the Examiner's rejection, but merely to advance prosecution of the instant application, Applicants respectfully submit that independent claim 31 has been amended to recite a combination of elements in a steering method including detecting a contact <u>by a contact detection sensor</u> between a part of said vehicle with a road surface side structure; and disengaging said clutch interposed therebetween in response to the contact. Applicants respectfully submit that this combination of elements as set forth in independent claim 31 is not disclosed or made obvious by the prior art of record, including Sadano and Tashiro.

The Examiner states that the combination of Sadano and Tashiro "discloses the steering system as described above" and therefore one presumably would have the same "expected result." Applicants respectfully submit that to the contrary, Sadano and Tashiro do not show or suggest the same steering system, since neither shows a steering feedback control at all, a ball screw axis, a nut connected to the ball screw axis, a clutch, or a link mechanism as claimed, and neither reference detects contact <u>by a contact detection sensor</u> between a part of said vehicle with a road surface side structure; and disengaging said clutch interposed therebetween in response to the contact. Applicants respectfully submit that the combination of elements as set forth in independent claim 31 is not disclosed or made obvious by the prior art of record, including Sadano and Tashiro, for the reasons explained above. Accordingly, reconsideration and withdrawal of this rejection are respectfully requested.

Claim 19 stands rejected under 35 U.S.C. § 103(a) as unpatentable over Sadano in view of Murata. This rejection is also respectfully traversed. As discussed in the Interview, Sadano fails to show or suggest said control section generates a provisional steering angle based on a current target deviation from a running route at a current position of said vehicle, a current actual deviation from said running route at said current position of said vehicle, and a current target steering angle at said current position of said vehicle, and optimize said provisional target

CG/PTS/py

steering angle to a control steering angle to minimize vibration of said vehicle resulting from a steering of said vehicle. Murata was cited to show conventional steering structure, but fails to show or suggest a control section that generates a provisional steering angle based on a current target deviation from a running route at a current position of said vehicle, a current actual deviation from said running route at said current position of said vehicle, and a current target steering angle at said current position of said vehicle, and optimize said provisional target steering angle to a control steering angle to minimize vibration of said vehicle resulting from a steering of said vehicle, and therefore cannot remedy the defects of Sadano. Reconsideration and withdrawal of this rejection are respectfully requested.

Conclusion

All objections and rejections raised in the Office Action having been properly traversed and addressed, it is respectfully submitted that the present application is in condition for allowance. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding rejections and that they be withdrawn. It is believed that a full and complete response has been made to the outstanding Office Action, and as such, the present application is in condition for allowance. Notice of same is earnestly solicited.

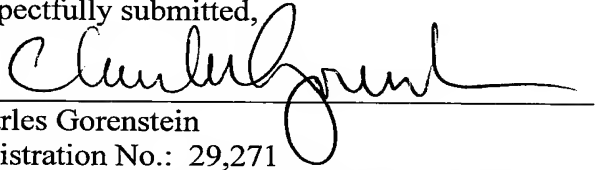Prompt and favorable consideration of this Amendment is respectfully requested.

If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is invited to telephone Paul T. Sewell, Registration No. 61,784, at (703) 205-8000, in the Washington, D.C. area.

CG/PTS/py

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No. 02-2448 for any additional fees required under 37 C.F.R. §§ 1.16 or 1.14; particularly, extension of time fees.

Dated: September 29, 2009

Respectfully submitted,

By _____
Charles Gorenstein
Registration No.: 29,271
BIRCH, STEWART, KOLASCH & BIRCH, LLP
8110 Gatehouse Road
Suite 100 East
P.O. Box 747
Falls Church, Virginia 22040-0747
(703) 205-8000
Attorney for Applicant

Attachment: 14 Pages of Wikipedia background information on "Genetic Algorithm"

CG/PTS/py

Genetic algorithm

From Wikipedia, the free encyclopedia

A **genetic algorithm (GA)** is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

Contents

Methodology

Genetic algorithms are implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The *fitness* of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly, then improve it through repetitive application of mutation, crossover, inversion and selection operators.

Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the *search space*). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection

Main article: Selection (genetic algorithm)

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a *fitness-based* process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

Reproduction

Main articles: crossover (genetic algorithm) and mutation (genetic algorithm)

The next step is to generate a second generation population of solutions from those selected through genetic

operators: crossover (also called recombination), and/or mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", recent researches (Islam Abou El Ata 2006) suggested more than two "parents" are better to be used to reproduce a good quality chromosome.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

Simple generational genetic algorithm pseudocode

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination: (time limit, sufficient fitness achieved, etc.)
    1. Select the best-fit individuals for reproduction
    2. Breed new individuals through crossover and mutation operations to give birth to offspring
    3. Evaluate the individual fitness of new individuals
    4. Replace least-fit population with new individuals

Observations

There are several general observations about the generation of solutions via a genetic algorithm:

- Repeated fitness function evaluation for complex problems is often the most prohibitive and limiting segment of artificial evolutionary algorithms. Finding optimal solution to complex high dimensional, multimodal problems often requires very expensive fitness function evaluations. In real world problems such as structural optimization problems, one single function evaluation may require several hours to several days of complete simulation. Typical optimization method can not deal with such a type of problem. In this case, it may be necessary to forgo an exact evaluation and use an approximated fitness that is computationally efficient. It is apparent that amalgamation of approximate models may be one of the most promising approaches to convincingly use EA to solve complex real life problems.
- The "better" is only in comparison to other solution. As a result, the stop criterion is not clear.
- In many problems, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. This means that it does not "know how" to sacrifice short-

term fitness to gain longer-term fitness. The likelihood of this occurring depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum, others may make it easier for the function to find the local optima. This problem may be alleviated by using a different fitness function, increasing the rate of mutation, or by using selection techniques that maintain a diverse population of solutions, although the No Free Lunch theorem proves that there is no general solution to this problem. A common technique to maintain diversity is to impose a "niche penalty", wherein, any group of individuals of sufficient similarity (niche radius) have a penalty added, which will reduce the representation of that group in subsequent generations, permitting other (less similar) individuals to be maintained in the population. This trick, however, may not be effective, depending on the landscape of the problem. Diversity is important in genetic algorithms (and genetic programming) because crossing over a homogeneous population does not yield new solutions. In evolution strategies and evolutionary programming, diversity is not essential because of a greater reliance on mutation.

- Operating on dynamic data sets is difficult, as genomes begin to converge early on towards solutions which may no longer be valid for later data. Several methods have been proposed to remedy this by increasing genetic diversity somehow and preventing early convergence, either by increasing the probability of mutation when the solution quality drops (called *triggered hypermutation*), or by occasionally introducing entirely new, randomly generated elements into the gene pool (called *random immigrants*). Again, evolution strategies and evolutionary programming can be implemented with a so-called "comma strategy" in which parents are not maintained and new parents are selected only from offspring. This can be more effective on dynamic problems.

- GAs cannot effectively solve problems in which the only fitness measure is a single right/wrong measure (like decision problems), as there is no way to converge on the solution (no hill to climb). In these cases, a random search may find a solution as quickly as a GA. *However*, if the situation allows the success/failure trial to be repeated giving (possibly) different results, then the ratio of successes to failures provides a suitable fitness measure.

- Selection is clearly an important genetic operator, but opinion is divided over the importance of crossover versus mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation, and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic). There are many references in Fogel (2006) that support the importance of mutation-based search, but across all problems the No Free Lunch theorem holds, so these opinions are without merit unless the discussion is restricted to a particular problem.

- Often, GAs can rapidly locate *good* solutions, even for difficult search spaces. The same is of course also true for evolution strategies and evolutionary programming.

- For specific optimization problems and problem instances, other optimization algorithms may find better solutions than genetic algorithms (given the same amount of computation time). Alternative and complementary algorithms include evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and swarm intelligence (e.g.: ant colony optimization, particle swarm optimization) and methods based on integer linear programming. The question of which, if any, problems are suited to genetic algorithms (in the sense that such algorithms are better than others) is open and controversial.

- As with all current machine learning problems it is worth tuning the parameters such as mutation probability, recombination probability and population size to find reasonable settings for the problem class being worked on. A very small mutation rate may lead to genetic drift (which is non-ergodic in nature). A recombination rate that is too high may lead to premature convergence of the genetic algorithm. A mutation rate that is too high may lead to loss of good solutions unless there is elitist selection. There are theoretical but not yet practical upper and lower bounds for these parameters that can help guide selection.

- The implementation and evaluation of the fitness function is an important factor in the speed and efficiency of the algorithm.

Variants

The simplest algorithm represents each chromosome as a bit string. Typically, numeric parameters can be represented by integers, though it is possible to use floating point representations. The floating point representation is natural to evolution strategies and evolutionary programming. The notion of real-valued genetic algorithms has been offered but is really a misnomer because it does not really represent the building block theory that was proposed by Holland in the 1970s. This theory is not without support though, based on theoretical and experimental results (see below). The basic algorithm performs crossover and mutation at the bit level. Other variants treat the chromosome as a list of numbers which are indexes into an instruction table, nodes in a linked list, hashes, objects, or any other imaginable data structure. Crossover and mutation are performed so as to respect data element boundaries. For most data types, specific variation operators can be designed. Different chromosomal data types seem to work better or worse for different specific problem domains.

When bit strings representations of integers are used, Gray coding is often employed. In this way, small changes in the integer can be readily effected through mutations or crossovers. This has been found to help prevent premature convergence at so called *Hamming walls*, in which too many simultaneous mutations (or crossover events) must occur in order to change the chromosome to a better solution.

Other approaches involve using arrays of real-valued numbers instead of bit strings to represent chromosomes. Theoretically, the smaller the alphabet, the better the performance, but paradoxically, good results have been obtained from using real-valued chromosomes.

A very successful (slight) variant of the general process of constructing a new population is to allow some of the better organisms from the current generation to carry over to the next, unaltered. This strategy is known as *elitist selection*.

Parallel implementations of genetic algorithms come in two flavours. Coarse grained parallel genetic algorithms assume a population on each of the computer nodes and migration of individuals among the nodes. Fine grained parallel genetic algorithms assume an individual on each processor node which acts with neighboring individuals for selection and reproduction. Other variants, like genetic algorithms for online optimization problems, introduce time-dependence or noise in the fitness function.

It can be quite effective to combine GA with other optimization methods. GA tends to be quite good at finding generally good global solutions, but quite inefficient at finding the last few mutations to find the absolute optimum. Other techniques (such as simple hill climbing) are quite efficient at finding absolute optimum in a limited region. Alternating GA and hill climbing can improve the efficiency of GA while overcoming the lack of robustness of hill climbing.

This means that the rules of genetic variation may have a different meaning in the natural case. For instance - provided that steps are stored in consecutive order - crossing over may sum a number of steps from maternal DNA adding a number of steps from paternal DNA and so on. This is like adding vectors that more probably may follow a ridge in the phenotypic landscape. Thus, the efficiency of the process may be increased by many orders of magnitude. Moreover, the inversion operator has the opportunity to place steps in consecutive order or any other suitable order in favour of survival or efficiency. (See for instance [1] or example in travelling salesman problem.)

Population-based incremental learning is a variation where the population as a whole is evolved rather than its individual members.

Problem domains

Problems which appear to be particularly appropriate for solution by genetic algorithms include timetabling and scheduling problems, and many scheduling software packages are based on GAs. GAs have also been applied to engineering. Genetic algorithms are often applied as an approach to solve global optimization problems.

As a general rule of thumb genetic algorithms might be useful in problem domains that have a complex fitness landscape as recombination is designed to move the population away from local optima that a traditional hill climbing algorithm might get stuck in.

History

Computer simulations of evolution started as early as in 1954 with the work of Nils Aall Barricelli, who was using the computer at the Institute for Advanced Study in Princeton, New Jersey.[2][3] His 1954 publication was not widely noticed. Starting in 1957,[4] the Australian quantitative geneticist Alex Fraser published a series of papers on simulation of artificial selection of organisms with multiple loci controlling a measurable trait. From these beginnings, computer simulation of evolution by biologists became more common in the early 1960s, and the methods were described in books by Fraser and Burnell (1970)[5] and Crosby (1973).[6] Fraser's simulations included all of the essential elements of modern genetic algorithms. In addition, Hans Bremermann published a series of papers in the 1960s that also adopted a population of solution to optimization problems, undergoing recombination, mutation, and selection. Bremermann's research also included the elements of modern genetic algorithms. Other noteworthy early pioneers include Richard Friedberg, George Friedman, and Michael Conrad. Many early papers are reprinted by Fogel (1998).[7]

Although Barricelli, in work he reported in 1963, had simulated the evolution of ability to play a simple game, [8] artificial evolution became a widely recognized optimization method as a result of the work of Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and early 1970s - Rechenberg's group was able to solve complex engineering problems through evolution strategies.[9][10][11][12] Another approach was the evolutionary programming technique of Lawrence J. Fogel, which was proposed for generating artificial intelligence. Evolutionary programming originally used finite state machines for predicting environments, and used variation and selection to optimize the predictive logics. Genetic algorithms in particular became popular through the work of John Holland in the early 1970s, and particularly his book *Adaptation in Natural and Artificial Systems* (1975). His work originated with studies of cellular automata, conducted by Holland and his students at the University of Michigan. Holland introduced a formalized framework for predicting the quality of the next generation, known as Holland's Schema Theorem. Research in GAs remained largely theoretical until the mid-1980s, when The First International Conference on Genetic Algorithms was held in Pittsburgh, Pennsylvania.

As academic interest grew, the dramatic increase in desktop computational power allowed for practical application of the new technique. In the late 1980s, General Electric started selling the world's first genetic algorithm product, a mainframe-based toolkit designed for industrial processes. In 1989, Axcelis, Inc. released Evolver, the world's second GA product and the first for desktop computers. The New York Times technology writer John Markoff wrote[13] about Evolver in 1990.

Related techniques

- Ant colony optimization (ACO) uses many ants (or agents) to traverse the solution space and find locally productive areas. While usually inferior to genetic algorithms and other forms of local search, it is able to produce results in problems where no global or up-to-date perspective can be obtained, and thus the other methods cannot be applied.

- Bacteriologic algorithms (BA) inspired by evolutionary ecology and, more particularly, bacteriologic adaptation. Evolutionary ecology is the study of living organisms in the context of their environment, with the aim of discovering how they adapt. Its basic concept is that in a heterogeneous environment, you can't find one individual that fits the whole environment. So, you need to reason at the population level. BAs have shown better results than GAs on problems such as complex positioning problems (antennas

for cell phones, urban planning, and so on) or data mining.[14]

- Cross-entropy method The cross-entropy (CE) method generates candidates solutions via a parameterized probability distribution. The parameters are updated via cross-entropy minimization, so as to generate better samples in the next iteration.

- Cultural algorithm (CA) consists of the population component almost identical to that of the genetic algorithm and, in addition, a knowledge component called the belief space.

- Evolution strategies (ES, see Rechenberg, 1994) evolve individuals by means of mutation and intermediate and discrete recombination. ES algorithms are designed particularly to solve problems in the real-value domain. They use self-adaptation to adjust control parameters of the search.

- Evolutionary programming (EP) involves populations of solutions with primarily mutation and selection and arbitrary representations. They use self-adaptation to adjust parameters, and can include other variation operations such as combining information from multiple parents.

- Extremal optimization (EO) Unlike GAs, which work with a population of candidate solutions, EO evolves a single solution and makes local modifications to the worst components. This requires that a suitable representation be selected which permits individual solution components to be assigned a quality measure ("fitness"). The governing principle behind this algorithm is that of *emergent* improvement through selectively removing low-quality components and replacing them with a randomly selected component. This is decidedly at odds with a GA that selects good solutions in an attempt to make better solutions.

- Gaussian adaptation (normal or natural adaptation, abbreviated NA to avoid confusion with GA) is intended for the maximisation of manufacturing yield of signal processing systems. It may also be used for ordinary parametric optimisation. It relies on a certain theorem valid for all regions of acceptability and all Gaussian distributions. The efficiency of NA relies on information theory and a certain theorem of efficiency. Its efficiency is defined as information divided by the work needed to get the information.[15] Because NA maximises mean fitness rather than the fitness of the individual, the landscape is smoothed such that valleys between peaks may disappear. Therefore it has a certain "ambition" to avoid local peaks in the fitness landscape. NA is also good at climbing sharp crests by adaptation of the moment matrix, because NA may maximise the disorder (average information) of the Gaussian simultaneously keeping the mean fitness constant.

- Genetic programming (GP) is a related technique popularized by John Koza in which computer programs, rather than function parameters, are optimized. Genetic programming often uses tree-based internal data structures to represent the computer programs for adaptation instead of the list structures typical of genetic algorithms.

- Grouping genetic algorithm (GGA) is an evolution of the GA where the focus is shifted from individual items, like in classical GAs, to groups or subset of items.[16] The idea behind this GA evolution proposed by Emanuel Falkenauer is that solving some complex problems, a.k.a. *clustering* or *partitioning* problems where a set of items must be split into disjoint group of items in an optimal way, would better be achieved by making characteristics of the groups of items equivalent to genes. These kind of problems include Bin Packing, Line Balancing, Clustering w.r.t. a distance measure, Equal Piles, etc., on which classic GAs proved to perform poorly. Making genes equivalent to groups implies chromosomes that are in general of variable length, and special genetic operators that manipulate whole groups of items. For Bin Packing in particular, a GGA hybridized with the Dominance Criterion of Martello and Toth, is arguably the best technique to date.

- Harmony search (HS) is an algorithm mimicking musicians behaviors in improvisation process.

- Interactive evolutionary algorithms are evolutionary algorithms that use human evaluation. They are usually applied to domains where it is hard to design a computational fitness function, for example, evolving images, music, artistic designs and forms to fit users' aesthetic preference.

- Memetic algorithm (MA), also called *hybrid genetic algorithm* among others, is a relatively new evolutionary method where local search is applied during the evolutionary cycle. The idea of memetic algorithms comes from memes, which unlike genes, can adapt themselves. In some problem areas they are shown to be more efficient than traditional evolutionary algorithms.

- Reactive search optimization (RSO) advocates the integration of sub-symbolic machine learning techniques into search heuristics for solving complex optimization problems. The word reactive hints at a ready response to events during the search through an internal online feedback loop for the self-tuning of critical parameters. Methodologies of interest for Reactive Search include machine learning and statistics, in particular reinforcement learning, active or query learning, neural networks, and meta-heuristics.

- Simulated annealing (SA) is a related global optimization technique that traverses the search space by testing random mutations on an individual solution. A mutation that increases fitness is always accepted. A mutation that lowers fitness is accepted probabilistically based on the difference in fitness and a decreasing temperature parameter. In SA parlance, one speaks of seeking the lowest energy instead of the maximum fitness. SA can also be used within a standard GA algorithm by starting with a relatively high rate of mutation and decreasing it over time along a given schedule.

- Stochastic optimization is an umbrella set of methods that includes GAs and numerous other approaches.

- Tabu search (TS) is similar to simulated annealing in that both traverse the solution space by testing mutations of an individual solution. While simulated annealing generates only one mutated solution, tabu search generates many mutated solutions and moves to the solution with the lowest energy of those generated. In order to prevent cycling and encourage greater movement through the solution space, a tabu list is maintained of partial or complete solutions. It is forbidden to move to a solution that contains elements of the tabu list, which is updated as the solution traverses the solution space.

Building block hypothesis

Genetic algorithms are relatively simple to implement, but their behavior is difficult to understand. In particular it is difficult to understand why they are often successful in generating solutions of high fitness. The building block hypothesis (BBH) consists of:

1. A description of an abstract adaptive mechanism that performs adaptation by recombining "building blocks", i.e. low order, low defining-length schemata with above average fitness.
2. A hypothesis that a genetic algorithm performs adaptation by implicitly and efficiently implementing this abstract adaptive mechanism.

(Goldberg 1989:41) describes the abstract adaptive mechanism as follows:

> Short, low order, and highly fit schemata are sampled, recombined [crossed over], and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata [the building blocks], we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.

> Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood [building blocks], so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemata, or building blocks.

(Goldberg 1989) claims that the building block hypothesis is supported by Holland's schema theorem.

The building block hypothesis has been sharply criticized on the grounds that it lacks theoretical justification and experimental results have been published that draw its veracity into question. On the theoretical side, for example, Wright et al. state that

> "The various claims about GAs that are traditionally made under the name of the *building block hypothesis* have, to date, no basis in theory and, in some cases, are simply incoherent"[17]

On the experimental side uniform crossover was seen to outperform one-point and two-point crossover on many of the fitness functions studied by Syswerda.[18] Summarizing these results, Fogel remarks that

> "Generally, uniform crossover yielded better performance than two-point crossover, which in turn yielded better performance than one-point crossover"[19]

Syswerda's results contradict the building block hypothesis because uniform crossover is extremely disruptive of short schemata whereas one and two-point crossover are more likely to conserve short schemata and combine their defining bits in children produced during recombination.

The debate over the building block hypothesis demonstrates that the issue of how GAs "work", (i.e. perform adaptation) is currently far from settled.

See also

- Algorithmic efficiency
- Holland's schema theorem
- Genetic programming
- Fitness approximation

Applications

- Artificial creativity
- Automated design, including research on composite material design and multi-objective design of automotive components for crashworthiness, weight savings, and other characteristics.
- Automated design of mechatronic systems using bond graphs and genetic programming (NSF).
- Automated design of industrial equipment using catalogs of exemplar lever patterns.
- Automated design of sophisticated trading systems in the financial sector.
- Building phylogenetic trees.[20]
- Calculation of bound states and local-density approximations.
- Chemical kinetics (gas (http://www.personal.leeds.ac.uk/~fuensm/project.html) and solid (http://repositories.cdlib.org/postprints/1154) phases)
- Configuration applications, particularly physics applications of optimal molecule configurations for particular systems like C60 (buckyballs).
- Container loading optimization.
- Code-breaking, using the GA to search large solution spaces of ciphers for the one correct decryption.[21]
- Design of water distribution systems.
- Distributed computer network topologies.

- Electronic circuit design, known as Evolvable hardware.
- File allocation for a distributed system.
- Game Theory Equilibrium Resolution.
- Gene expression profiling analysis.[22]
- Genetic Algorithm for Rule Set Production
- Learning Robot behavior using Genetic Algorithms.
- Learning fuzzy rule base using genetic algorithms.
- Linguistic analysis, including Grammar induction and other aspects of Natural language processing (NLP) such as word sense disambiguation.
- Marketing Mix Analysis
- Mobile communications infrastructure optimization.
- Molecular Structure Optimization (Chemistry).
- Multiple criteria production scheduling.[23]
- Multiple population topologies and interchange methodologies.
- Mutation testing
- Neural Networks; particularly recurrent neural networks[24]
- Operon prediction.[25]
- Optimisation of data compression systems, for example using wavelets.
- Parallelization of GAs/GPs including use of hierarchical decomposition of problem domains and design spaces nesting of irregular shapes using feature matching and GAs.
- Pop music record producer.[26]
- Protein folding and protein/ligand docking.[27]
- Plant floor layout.
- Representing rational agents in economic models such as the cobweb model.
- Bioinformatics: RNA structure prediction.[28]

- Bioinformatics: [Multiple Sequence Alignment].[29] SAGA is available on:[30] .

- Bioinformatics Multiple sequence alignment.[31]

- Scheduling applications, including job-shop scheduling. The objective being to schedule jobs in a sequence dependent or non-sequence dependent setup environment in order to maximize the volume of production while minimizing penalties such as tardiness.
- Selection of optimal mathematical model to describe biological systems.
- Software engineering
- Solving the machine-component grouping problem required for cellular manufacturing systems.
- Tactical asset allocation and international equity strategies.
- Timetabling problems, such as designing a non-conflicting class timetable for a large university.
- Training artificial neural networks when pre-classified training examples are not readily obtainable (neuroevolution).
- Traveling Salesman Problem.
- Finding hardware bugs.[32][33]
- Wireless Sensor/Ad-hoc Networks.[34]
- Data Center/Server Farm.[35]

Notes

1. ^ Evolution-in-a-nutshell (http://web.telia.com/~u91131915/traveller.htm)
2. ^ Barricelli, Nils Aall (1954). "Esempi numerici di processi di evoluzione". *Methodos*: 45–68.
3. ^ Barricelli, Nils Aall (1957). "Symbiogenetic evolution processes realized by artificial methods". *Methodos*: 143–182.

4. ^ Fraser, Alex (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction". *Aust. J. Biol. Sci.* **10**: 484–491.
5. ^ Fraser, Alex; Donald Burnell (1970). *Computer Models in Genetics*. New York: McGraw-Hill.
6. ^ Crosby, Jack L. (1973). *Computer Simulation in Genetics*. London: John Wiley & Sons.
7. ^ Fogel, David B. (editor) (1998). *Evolutionary Computation: The Fossil Record*. New York: IEEE Press.
8. ^ Barricelli, Nils Aall (1963). "Numerical testing of evolution theories. Part II. Preliminary tests of performance, symbiogenesis and terrestrial life". *Acta Biotheoretica* (16): 99–126.
9. ^ Rechenberg, Ingo (1973). *Evolutionsstrategie*. Stuttgart: Holzmann-Froboog.
10. ^ Schwefel, Hans-Paul (1974). *Numerische Optimierung von Computer-Modellen (PhD thesis)*.
11. ^ Schwefel, Hans-Paul (1977). *Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie : mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*. Basel; Stuttgart: Birkhäuser. ISBN 3764308761.
12. ^ Schwefel, Hans-Paul (1981). *Numerical optimization of computer models (Translation of 1977 Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie*. Chichester ; New York: Wiley. ISBN 0471099880.
13. ^ Markoff, John (1989). "What's the Best Answer? It's Survival of the Fittest (http://www.nytimes.com/1990/08/29/business/business-technology-what-s-the-best-answer-it-s-survival-of-the-fittest.html)". New York Times. http://www.nytimes.com/1990/08/29/business/business-technology-what-s-the-best-answer-it-s-survival-of-the-fittest.html. Retrieved 2009-08-09.
14. ^ Baudry, Benoit; Franck Fleurey, Jean-Marc Jézéquel, and Yves Le Traon (March/April 2005). "Automatic Test Case Optimization: A Bacteriologic Algorithm (http://www.irisa.fr/triskell/publis/2005/Baudry05d.pdf)" (PDF). *IEEE Software* (IEEE Computer Society) **22**: 76–82. doi:10.1109/MS.2005.30 (http://dx.doi.org/10.1109/MS.2005.30). http://www.irisa.fr/triskell/publis/2005/Baudry05d.pdf. Retrieved 2009-08-09.
15. ^ Kjellström, G. (December 1991). "On the Efficiency of Gaussian Adaptation". *Journal of Optimization Theory and Applications* **71** (3): 589–597. doi:10.1007/BF00941405 (http://dx.doi.org/10.1007/BF00941405).
16. ^ Falkenauer, Emanuel (1997). *Genetic Algorithms and Grouping Problems*. Chichester, England: John Wiley & Sons Ltd. ISBN 978-0-471-97150-4.
17. ^ Wright, A.H.; et al. (2003). "Implicit Parallelism". Proceedings of the Genetic and Evolutionary Computation Conference.
18. ^ Syswerda, G. (1989). "Uniform crossover in genetic algorithms". in J. D. Schaffer. Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann.
19. ^ Fogel, David B. (2000). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. New York: IEEE Press. pp. 140.
20. ^ Hill T, Lundgren A, Fredriksson R, Schiöth HB (2005). "Genetic algorithm for large-scale maximum parsimony phylogenetic analysis of proteins". *Biochimica et Biophysica Acta* **1725**: 19–29. PMID 15990235 (http://www.ncbi.nlm.nih.gov/pubmed/15990235).
21. ^ Joachim De Zutter
22. ^ To CC, Vohradsky J (2007). "A parallel genetic algorithm for single class pattern classification and its application for gene expression profiling in Streptomyces coelicolor". *BMC Genomics* **8**: 49. doi:10.1186/1471-2164-8-49 (http://dx.doi.org/10.1186/1471-2164-8-49). PMID 17298664 (http://www.ncbi.nlm.nih.gov/pubmed/17298664).
23. ^ Bagchi Tapan P (1999). *Multiobjective Scheduling by Genetic Algorithms*.
24. ^ Applying Genetic Algorithms to Recurrent Neural Networks for Learning Network Parameters and Architecture (http://arimaa.com/arimaa/about/Thesis/)
25. ^ Wang S, Wang Y, Du W, Sun F, Wang X, Zhou C, Liang Y (2007). "A multi-approaches-guided genetic algorithm with application to operon prediction". *Artificial Intelligence in Medicine* **41**: 151–159. doi:10.1016/j.artmed.2007.07.010 (http://dx.doi.org/10.1016/j.artmed.2007.07.010). PMID 17869072 (http://www.ncbi.nlm.nih.gov/pubmed/17869072).
26. ^ BBC News | Entertainment | To the beat of the byte (http://news.bbc.co.uk/2/hi/entertainment/123983.stm)
27. ^ Willett P (1995). "Genetic algorithms in molecular recognition and design". *Trends in Biotechnology* **13**: 516–521. doi:10.1016/S0167-7799(00)89015-0 (http://dx.doi.org/10.1016/S0167-7799(00)89015-0). PMID 8595137 (http://www.ncbi.nlm.nih.gov/pubmed/8595137).
28. ^ van Batenburg FH, Gultyaev AP, Pleij CW (1995). "An APL-programmed genetic algorithm for the prediction of RNA secondary structure". *Journal of Theoretical Biology* **174**: 269–280. doi:10.1006/jtbi.1995.0098 (http://dx.doi.org/10.1006/jtbi.1995.0098). PMID 7545258 (http://www.ncbi.nlm.nih.gov/pubmed/7545258).
29. ^ Notredame C, Higgins DG (1995). "SAGA a Genetic Algorithm for Multiple Sequence Alignment". *Nulceic Acids Research* **174**: 1515. PMID 8628686 (http://www.ncbi.nlm.nih.gov/pubmed/8628686).
30. ^ Cedric Notredame Home Page (http://www.tcoffee.org/homepage.html)
31. ^ Gondro C, Kinghorn BP (2007). "A simple genetic algorithm for multiple sequence alignment". *Genetics and Molecular Research* **6**: 964–982. PMID 18058716 (http://www.ncbi.nlm.nih.gov/pubmed/18058716).

32. ^ Hitoshi Iba, Sumitaka Akiba, Tetsuya Higuchi, Taisuke Sato: BUGS: A Bug-Based Search Strategy using Genetic Algorithms. PPSN 1992:
33. ^ Ibrahim, W. and Amer, H.: An Adaptive Genetic Algorithm for VLSI Test Vector Selection
34. ^ BiSNET/e - Distributed Software Systems Group, University of Massachusetts, Boston (http://dssg.cs.umb.edu/wiki/index.php/BiSNET/e)
35. ^ SymbioticSphere - Distributed Software Systems Group, University of Massachusetts, Boston (http://dssg.cs.umb.edu/wiki/index.php/SymbioticSphere)

References

- Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank (1998) *Genetic Programming - An Introduction*, Morgan Kaufmann, San Francisco, CA.
- Bies, Robert R; Muldoon, Matthew F; Pollock, Bruce G; Manuck, Steven; Smith, Gwenn and Sale, Mark E (2006). "A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection". *Journal of Pharmacokinetics and Pharmacodynamics* (Netherlands: Springer): 196–221.
- Cha, Sung-Hyuk; Tappert, Charles C (2009). "A Genetic Algorithm for Constructing Compact Binary Decision Trees (http://www.jprr.org/index.php/jprr/article/view/44/25)". *Journal of Pattern Recognition Research (http://www.jprr.org/index.php/jprr)* 4 (1): 1–13.
- Fraser, Alex S. (1957). "Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction". *Australian Journal of Biological Sciences* 10: 484–491.
- Goldberg, David E (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Kluwer Academic Publishers, Boston, MA.
- Goldberg, David E (2002), *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Addison-Wesley, Reading, MA.
- Fogel, David B (2006), *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ. Third Edition
- Holland, John H (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor
- Koza, John (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press. ISBN 0-262-11170-5
- Michalewicz, Zbigniew (1999), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag.
- Mitchell, Melanie, (1996), *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.
- Poli, R., Langdon, W. B., McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu.com, freely available from the internet. ISBN 978-1-4092-0073-4.
- Rechenberg, Ingo (1994): Evolutionsstrategie '94, Stuttgart: Fromman-Holzboog.
- Schmitt, Lothar M; Nehaniv, Chrystopher L; Fujii, Robert H (1998), *Linear analysis of genetic algorithms*, Theoretical Computer Science 208: 111-148
- Schmitt, Lothar M (2001), *Theory of Genetic Algorithms*, Theoretical Computer Science 259: 1-61
- Schmitt, Lothar M (2004), *Theory of Genetic Algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling*, Theoretical Computer Science 310: 181-231
- Schwefel, Hans-Paul (1974): Numerische Optimierung von Computer-Modellen (PhD thesis). Reprinted by Birkhäuser (1977).
- Vose, Michael D (1999), *The Simple Genetic Algorithm: Foundations and Theory*, MIT Press, Cambridge, MA.
- Whitley, D. (1994). *A genetic algorithm tutorial*. Statistics and Computing 4, 65–85.

External links

- [1] (http://twtmas.mpei.ac.ru/mas/Worksheets/Minimum.mcd) Search of Global Minimum by genetic algorithme

An Alternative to the Building Block Hypothesis

- Generative Fixation (http://cs.brandeis.edu/~kekib/dissertation.html) A unified explanation for the adaptive capacity of simple recombinative genetic algorithms

Applications

- Demo applet of a evolutionary algorithm for solving TSP's and VRPTW problems (http://www.dna-evolutions.com/dnaappletsample.html)
- Genetic Arm (http://www.e-nuts.net/en/genetic-algorithms) Simulation of a mechanical arm trained using genetic algorithms. Custom goals can be defined using a scripting language. A sample video is available on page.
- Antenna optimization for NASA (http://ti.arc.nasa.gov/projects/esg/research/antenna.htm) A successful application of genetic algorithms.
- Genesis-SGA Seo genetic Algorithm (http://seo.witinside.net/genetic-algorithms/) Genetic algorithms applied to the theme SEO (Search Engine Optimization)

Resources

- DigitalBiology.NET (http://www.digitalbiology.net/) Vertical search engine for GA/GP resources
- Genetic Algorithms Index (http://www.geneticprogramming.com/ga/index.htm) The site Genetic Programming Notebook provides a structured resource pointer to web pages in genetic algorithms field

Tutorials

- A Field Guide to Genetic Programming (http://www.gp-field-guide.org.uk/) A book, freely downloadable under a Creative Commons license.
- Introduction to Genetic Algorithms with interactive Java applets (http://www.obitko.com/tutorials/genetic-algorithms/) For experimenting with GAs online
- A Practical Tutorial on Genetic Algorithm (http://fog.neopages.org/helloworldgeneticalgorithms.php) Programming a Genetic Algorithm step by step.
- A Genetic Algorithm Tutorial by Darrell Whitley Computer Science Department Colorado State University (http://samizdat.mines.edu/ga_tutorial/ga_tutorial.ps) An excellent tutorial with lots of theory
- Cross discipline example applications for GAs with references. (http://www.talkorigins.org/faqs/genalg/genalg.html)
- Global Optimization Algorithms - Theory and Application (http://www.it-weise.de/projects/book.pdf)

Libraries

- ECJ (http://cs.gmu.edu/~eclab/projects/ecj/), among the most popular Java evolutionary computation libraries.
- EO (http://eodev.sourceforge.net/), a very popular C++ evolutionary computation toolkit.
- ParadisEO (http://paradiseo.gforge.inria.fr/), a parallel and multiobjective extension to EO (http://eodev.sourceforge.net/).
- Open BEAGLE (http://beagle.gel.ulaval.ca/), another popular C++ evolutionary computation toolkit.
- Demo applet of JOpt.SDK (http://www.dna-evolutions.com/dnaappletsample.html) an evolutionary algorithm software library for Java or .NET for solving TSP's and VRPTW problems
- Evoptool (http://airwiki.elet.polimi.it/mediawiki/index.php/Evoptool:_Evolutive_Optimization_Tool) A framework and a set of libraries written in C++ for the Evolutive Computation, including several Genetic Algorithms and EDAs.
- Jenes (http://sites.google.com/a/ciselab.org/jenes/) An optimized Java library for Genetic Algorithms.
- Pyevolve (http://pyevolve.sourceforge.net/) A python framework for Genetic Algorithms.
- Genetic Algorithms in Ruby (http://ai4r.rubyforge.org/geneticAlgorithms.html)
- GAlib (http://lancet.mit.edu/ga/) A C++ Library of Genetic Algorithm Components
- MOGAlib (http://mogalib.uni-pannon.hu/) Multi-objective extension to the GALib C++ Library
- GAEDALib (http://laurel.datsi.fi.upm.es/projects/gaedalib) A C++ Library of Evolutive Algotithms (GAs, EDAs, DEs and others) based in GAlib, and supporting to MOS and parallel computing
- Jenetics (http://jenetics.sourceforge.net/) Genetic Algorithm Library written in Java.

- ga (http://www.mathworks.com/access/helpdesk/help/toolbox/gads/ga.html) Genetic Algorithm in MATLAB (How GA in MATLAB works (http://www.mathworks.com/access/helpdesk/help/toolbox/gads/index.html?/access/helpdesk/help/toolbox
- gamultiobj (http://www.mathworks.com/access/helpdesk/help/toolbox/gads/gamultiobj.html) Multitobjective Genetic Algorithm in MATLAB
- GARAGe (http://garage.cse.msu.edu/) Michigan State University's Genetic Algorithm library in C, GALLOPS
- GAOT (http://www.ise.ncsu.edu/mirage/GAToolBox/gaot/) The Genetic Algorithm Optimization Toolbox (GAOT) for Matlab, by NCSU
- JGAP (http://jgap.sourceforge.net/) Java Genetic Algorithms Package features comprehensive unit tests
- speedyGA (http://blog.hackingevolution.net/2009/02/04/speedyga-v13/) A fast lightweight genetic algorithm in Matlab

Retrieved from "http://en.wikipedia.org/wiki/Genetic_algorithm"
Categories: Cybernetics | Intelligence | Genetic algorithms | Optimization algorithms | Search algorithms

- This page was last modified on 19 September 2009 at 21:08.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details.
  Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.